The MIp Toolset: an efficient algorithm for calculating Mutual Information in protein alignments

Russell J Dickson and Gregory B Gloor[*]
Department of Biochemistry, University of Western Ontario, London, ON, Canada

Abstract

Background: Coevolution within a protein family is often predicted using statistics that measure the degree of covariation between positions in the protein sequence. Mutual Information is a measure of dependence between two random variables that has been used extensively to predict intra-protein coevolution.

Results: Here we provide an algorithm for the efficient calculation of Mutual Information within a protein family. The algorithm uses linked lists which are directly accessed by a pointer array. The linked list allows efficient storage of sparse count data caused by protein conservation. The direct access array of pointers prevents the linked list from being traversed each time it is modified.

Conclusions: This algorithm is implemented in the software MIp-Toolset, but could also be easily implemented in other Mutual Information based standalone software or web servers. The current implementation in the MIpToolset has been critical in large-scale protein family analysis and real-time coevolution calculations during alignment editing and curation.

The MIpToolset is available at:
https://sourceforge.net/projects/miptoolset/

Introduction

The identification and analysis of covarying positions in a protein family gives important insights into that family's evolutionary history and provides information about sites that are important for function and structural stability as it is believed that covariation implies coevolution [1, 2, 3, 4]. Coevolutionary analysis of protein families is important because it potentially provides a direct link between primary sequence, in the form of multiple sequence alignments, and structure/function predictions. Covariation between positions in a protein

[*]Corresponding Author: ggloor@uwo.ca

family is assumed to derive from phylogenetic, structural, functional, interaction, and stochastic signals [1]. Decomposing this signal is difficult because the phylogenetic and stochastic signal can overwhelm the structural and functional signal [5]. Furthermore, alignment errors have been shown to produce misleading erroneous signal [6].

One of the most popular methods for quantifying covariation in proteins is Mutual Information (MI). There are many coevolution prediction methods which are derived from MI [3, 7, 8, 9, 6]. As well, there are many web-based servers which will calculate Mutual Information from a submitted protein alignment [10, 11, 12, 13]. Despite its simple formulation, calculation of MI is computationally demanding, largely because it must be calculated for all pairs of positions in the alignment, meaning it scales n^2 relative to the length of the alignment. Further, calculating inter-protein coevolution requires concatenated alignments which increases the effective number of pairs of positions.

Herein we describe an algorithm for calculating MI in protein alignments with high efficiency. This algorithm allows for database-wide analysis [6] and real-time calculation of covariation during alignment curation [14]. This algorithm is included as part of the MIpToolset.

Algorithm

Mutual Information

The calculation and formulation of Mutual Information is described in detail in [5]; it is outlined here to provide necessary background to understand the optimizations of the MIpToolset algorithm.

Mutual Information measures the degree of covariation between two random variables (in our case, protein alignment positions X and Y) using the Information Theoretic quantity Entropy (H).

$$MI_{x,y} = H_x + H_y - H_{x,y} \tag{1}$$

Information Entropy (H) can be understood as the measure of uncertainty of the identity of the amino acid at some position x. As shown in equation 2, the Entropy (H) for position x is calculated using the probability of each of the 20 amino acids appearing at that position. Since the actual probabilities are unknown, the amino acid frequencies in the input alignment are used to approximate these values.

$$H_x = -\sum_{i=1}^{20} p(x_i) \log_{20} p(x_i) \tag{2}$$

The MI between positions X and Y is the sum of the Entropy of each position minus the "joint Entropy" between them. The enumeration of joint entropy is the rate-limiting step of Mutual Information calculations. Joint Entropy is calculated similarly to Entropy, but it involves the calculation of probability of all pairs of amino acids that occur between position x and position y (Equation 3).

$$H_{x,y} = -\sum_{i=1}^{20}\sum_{j=1}^{20} p(x_i, y_j) \log_{20} p(x_i, y_j) \tag{3}$$

The naïve calculation of joint entropy is inefficient because it involves populating a 20 x 20 matrix for every pair of amino acids found for every pair of positions. This is a 400-entry matrix for n^2 positions. This approach, while easy to implement, uses an unnecessary amount of memory as it does not exploit the fact that most positions will be moderately conserved and, thus, most positions will have a value of zero in the joint entropy count matrix.

Storage of sparse matrix in linked list

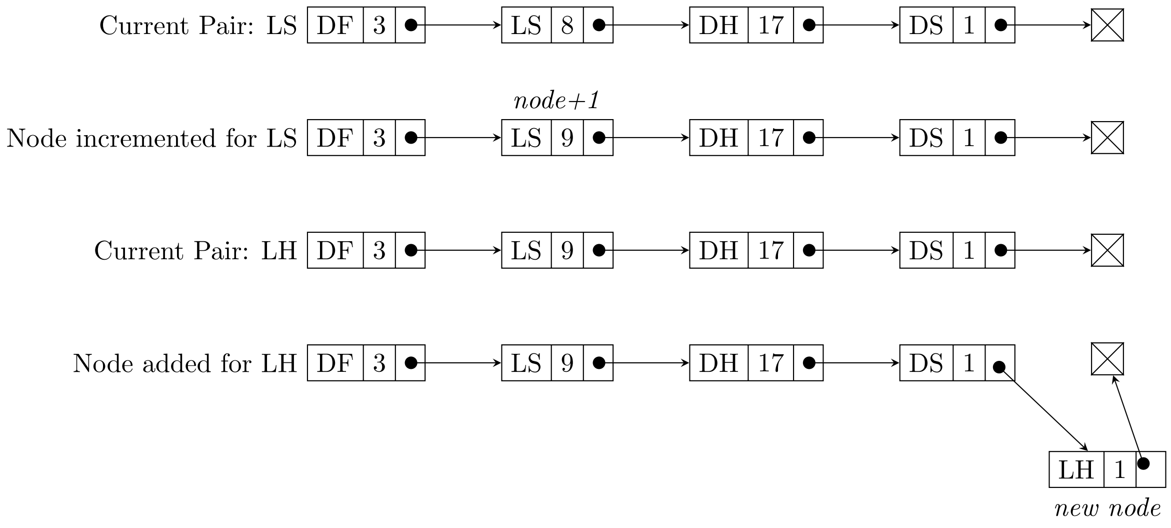

Figure 1: **The linked list storage of amino acid pair counts.** This figure demonstrates how two new amino acid pairs, LS and LH, are added to the growing linked list data structure which stores amino acid pair counts. First, LS is added to the list by incrementing the existing LS node. Second, the pair LH is added to the list by creating a new node labeled LH and adding it to the list with counter set to 1.

It is worth noting that calculation of MI involves two types of "pairs": Pairs of *positions*, which represent the homologous 'columns' in a protein family multiple sequence alignment (MSA), and pairs of *amino acids*, which are the corresponding entries from a pair of positions within a single sequence. So a pair of positions, might be position 10 and position 45 within a protein sequence; at this pair of positions, there will be many amino acid pairs corresponding to the identity of the amino acids at positions 10 and 45 in the sequence (ie. DF, LS, DH etc.).

A straightforward way to store the counts between positions x and y is to use a linked list data structure (Figure 1). Each node in the linked list stores two values for the calculation of Joint Entropy, the identity of the amino acid pair, and the respective count. Each node also contains a pointer to the next node in the list, or *null* if the node is the terminal node.

The program iterates over the protein alignment, enumerating the amino acid pairs, just as it would if it were in the naïve implementation. If an entry in the linked list exists for a given amino acid pair, the node's counter is incremented. If no such entry exists a new node is appended to the end of the list for that amino acid pair. This list can be traversed efficiently as these counts will be used for future calculations. This efficient storage makes it possible to efficiently analyze very long alignments.

Direct access to linked list improves speed

The limitation of the linked list storage method, if a linked list is used on its own, is that the list will need to be traversed each time a node is to be updated or created to check whether that pair exists in the data structure. This challenge can be overcome by using an array of pointers to linked list nodes. The disadvantage of the linked list storage solution is that it lacks "direct access" provided by a two-dimensional array in from the naïve implementation. By combining the two, it is possible to achieve a "best of both worlds" solution.

A single "direct-access" 20 x 20 array is created, with the nodes in the array corresponding to the 400 possible amino acid pairs (Figure 2). When an amino acid pair is encountered by the main count enumeration loop, the direct-access array is checked. If the entry for that pair is *null*, then a new linked node is appended to the end of the growing linked list for that pair of positions with a count of 1; next, the entry in the direct-access matrix is set as a pointer to the newly created linked list node.

Conversely, if the entry corresponding to the amino acid pair contains a pointer, the program follows the pointer to the corresponding linked list node and increments the counter by 1. After the two positions have been fully enumerated, all entries in the direct-access array are reset to *null* and it can be reused. Thus, the direct-access array strategy maintains the advantages of a linked list storage solution without the disadvantage of needing to traverse the list every time, at the trade-off cost of only 400 pointers.

Integration in the MIpToolset

This algorithm has been included as part of the MIpToolset, a collection of C- and Perl-based programs which calculate covariation statistics and inter-residue distances from protein alignments and databases. A full description of sequence

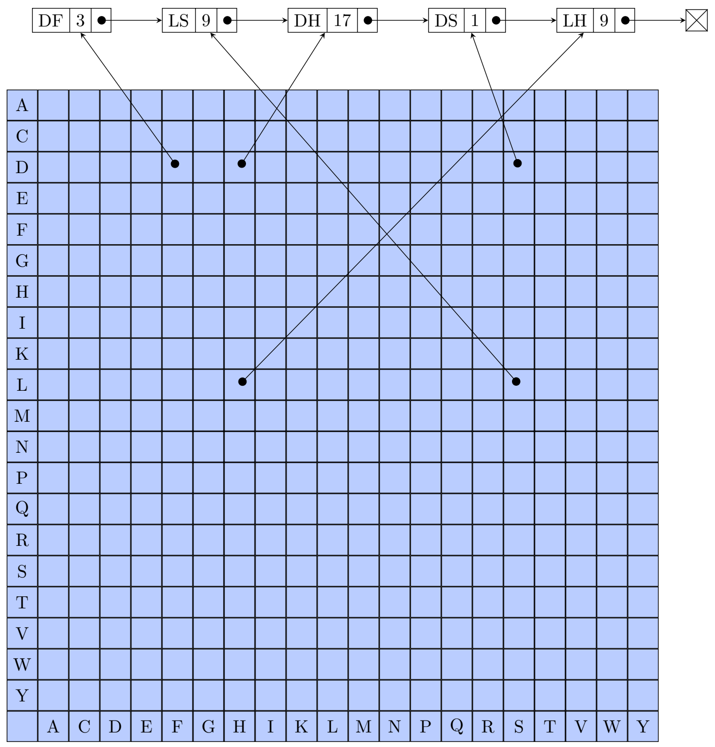
DF 3
LS 9
DH 17
DS 1
LH 9
A C D E F G H I K L M N P Q R S T V W Y
A C D E F G H I K L M N P Q R S T V W Y

Figure 2: **Direct-access array of pointers to growing linked list.** This figure demonstrates how the direct-access array provides instant access to any part of the linked list without the need to traverse the list. This array is only allocated once and can be reused for each pair of positions.

collection and alignment is available in (Dickson and Gloor, Methods Mol. Biol. 2013 *submitted*).

In brief, the input to the program is a protein alignment containing more than 150 sequences less than 90% identical and containing more than 50 ungapped positions. It is recommended that the alignment be manually analyzed by the investigator to ensure the alignment does not contain errors which will lead to false-positive results [6, 14]. For example, the curation tool LoCo [14], based on the alignment viewer Jalview [15], provides a visualization of the likely-misaligned regions of the alignment. The program also optionally accepts a PDB structure corresponding to a sequence in the protein family. This structure is used to generate inter-residue distances which are commonly used to validate coevolution predictions.

The output of the program is a large list of pairs of positions and their corresponding covariation statistics. The MIpToolset presently generates Mutual Information, as well as several more accurate derivations including MIp (and its normalized counterpart Zp) [7], Zpx and ΔZp [6]. A coevolution network file is also produced which can be visualized using Graphviz [16].

Conclusions

It is established that MI by itself is not particularly accurate in predicting coevolving positions because it correlates with Entropy [5], misleading phylogenetic signal [7], and alignment errors [6]. Furthermore, analyzing gaps as the "21st amino acid" causes misleading results which is partially why the aforementioned studies excluded positions containing gaps from the analysis (Dickson et al. submitted). It is possible to overcome some limitations of raw MI by using various corrections to MI [7, 8, 9, 6]. Typically these corrections based on an analysis of raw MI values are computationally inexpensive and so heavy optimization is not necessary. Thus the algorithm and software described herein can be used to reduce the time and memory required to calculate most MI-derived statistics.

The MIpToolset has been tested on Unix-like operating systems and is implemented in C for efficiency, with a Perl wrapper for handling input/output issues. The speed and efficiency of the MIpToolset has allowed for efficient database-wide analysis [6] and the detection of protein family misalignments using an MI-derived method in real-time as the user edits their alignment in the software tool LoCo [14]. To our knowledge, this is the fastest implementation of the coevolution statistics MIp, Zp, Zpx, and ΔZp[7, 6].

It is available at: https://sourceforge.net/projects/miptoolset/

Authors contributions

RJD designed the algorithm, and wrote the manuscript. GBG designed the project. All authors contributed to the software, and read and approved the final manuscript.

Acknowledgements

The authors wish to thank the students of the 2011 and 2012 Biochemistry 4445a class at UWO for identifying documentation ambiguities and cross-platform issues. Also, thank you to Stan Dunn, Lindi Wahl, David Edgell, Thomas McMurrough, Jean Macklaim, Andrew Fernandes, Ardeshir Goliaei for helpful discussions and testing. Finally thank you to existing users of the MIpToolset for their feedback while using this software while it was in development.

References

[1] Atchley WR, Wollenberg KR, Fitch WM, Terhalle W, Dress AW: **Correlations among amino acid sites in bHLH protein domains: an information theoretic analysis.** *Mol Biol Evol* 2000, **17**:164–178.

[2] Tillier E, Lui T: **Using multiple interdependency to separate functional from phylogenetic correlations in protein alignments**. *Bioinformatics* 2003, **19**(6):750–755.

[3] Gloor GB, Martin LC, Wahl LM, Dunn SD: **Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions.** *Biochemistry* 2005, **44**(19):7156–7165.

[4] Travers SAA, Fares MA: **Functional coevolutionary networks of the Hsp70-Hop-Hsp90 system revealed through computational analyses.** *Mol Biol Evol* 2007, **24**(4):1032–1044.

[5] Martin LC, Gloor GB, Dunn SD, Wahl LM: **Using information theory to search for co-evolving residues in proteins.** *Bioinformatics* 2005, **21**(22):4116–4124.

[6] Dickson R, Wahl L, Fernandes A, Gloor G: **Identifying and seeing beyond multiple sequence alignment errors using intra-molecular protein covariation**. *PLoS ONE* 2010, **5**(6):e11082.

[7] Dunn SD, Wahl LM, Gloor GB: **Mutual information without the influence of phylogeny or entropy dramatically improves residue contact prediction**. *Bioinformatics* 2008, **24**(3):333–340.

[8] Little DY, Chen L: **Identification of Coevolving Residues and Coevolution Potentials Emphasizing Structure, Bond Formation and Catalytic Coordination in Protein Evolution**. *PLoS ONE* 2009, **4**(3):e4762.

[9] Buslje CM, Santos J, Delfino JM, Nielsen M: **Correction for phylogeny, small number of observations and data redundancy improves the identification of coevolving amino acid pairs using mutual information.** *Bioinformatics* 2009, **25**(9):1125–1131.

[10] Kozma D, Simon I, Tusnády GE: **CMWeb: an interactive on-line tool for analysing residue-residue contacts and contact prediction methods.** *Nucleic Acids Res* 2012, **40**(Web Server issue):W329–33.

[11] Chakraborty A, Mandloi S, Lanczycki CJ, Panchenko AR, Chakrabarti S: **SPEER-SERVER: a web server for prediction of protein specificity determining sites.** *Nucleic Acids Res* 2012, **40**(Web Server issue):W242–8.

[12] Yip KY, Patel P, Kim PM, Engelman DM, McDermott D, Gerstein M: **An integrated system for studying residue coevolution in proteins.** *Bioinformatics* 2008, **24**(2):290–292.

[13] Gouveia-Oliveira R, Roque FS, Wernersson R, Sicheritz-Ponten T, Sackett PW, Mølgaard A, Pedersen AG: **InterMap3D: predicting and visualizing co-evolving protein residues.** *Bioinformatics* 2009, **25**(15):1963–1965.

[14] Dickson RJ, Gloor GB: **Protein sequence alignment analysis by local covariation: coevolution statistics detect benchmark alignment errors.** *PLoS ONE* 2012, **7**(6):e37645.

[15] Waterhouse AM, Procter JB, Martin DMA, Clamp M, Barton GJ: **Jalview Version 2–a multiple sequence alignment editor and analysis workbench**. *Bioinformatics* 2009, **25**(9):1189–1191.

[16] Ellson J, Gansner E, Koutsofios L, North SC, Woodhull G: **Graphviz—open source graph drawing tools**. *Lecture Notes in Computer Science* 2002, :483–484.